Cryptographic Protocols
Formal and Computational Proofs
Introduction

David Baelde

LSV, ENS Paris-Saclay

2019

(based on slides by B. Blanchet & S. Delaune)

Cryptographic protocols

Paiement Internet

Cryptographic protocols

Communicating processes that use cryptographic primitives to meet security properties in a hostile environment.

Security properties (1)

- Secrecy: May an intruder learn some secret message?
- Authentication: Is the agent Alice really talking to Bob?
- Non-repudiation: Alice sends a message to Bob. Alice cannot later deny having sent this message. Bob cannot deny having received the message.
- Forward security: Past communications remain secure if long-term keys are compromised.
- Post-compromise security: Future communications are secure soon after long-term keys are compromised.

Security properties: E-voting (2)

Eligibility: only legitimate voters can vote

Fairness: no early results can be obtained which could influence the remaining voters

Individual verifiability:
a voter can verify that her vote was really counted

Universal verifiability:
anyone can verify that the outcome really is the sum of all the votes

Belgique - Election 2004 - http://www.poureva.be/ - (C) Kanar

Security properties: E-voting (3)

Privacy: the fact that somebody voted in a particular way is not revealed to anyone

Receipt-freeness: a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)

Coercion-resistance: same as receipt-freeness, but the coercer interacts with the voter during the protocol, e.g. by preparing messages

Cryptographic primitives

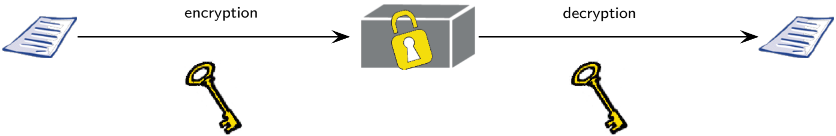
Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, encryption and signature functions.

Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, encryption and signature functions.

Symmetric encryption

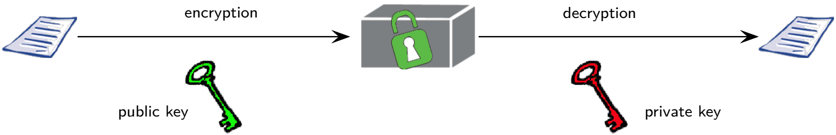


$\longrightarrow$ Examples: Caesar encryption, DES, AES, ...

Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, encryption and signature functions.

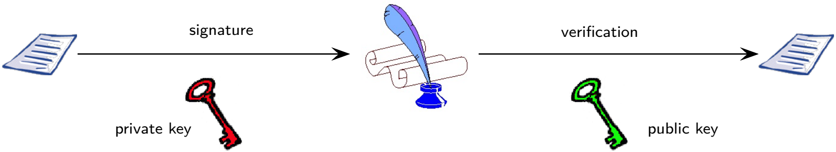
Asymmetric encryption



Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, encryption and signature functions.

Signature



Credit Card Payment Protocol

Example: credit card payment

- The client Cl puts his credit card C in the terminal T.
- The merchant enters the amount M of the sale.

- The terminal authenticates the credit card.
- The client enters his PIN.
  If $M \geq 100$€, then in 20% of cases,
  - The terminal contacts the bank B.
  - The bank gives its authorisation.

More details

The Bank B , the Client Cl, the Credit Card C and the Terminal T

More details

The Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

More details

The Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\{hash(Data)\}_{\mathrm{priv}(B)}$
- a secret key shared with the bank – K_{CB}

More details

The Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\{hash(Data)\}_{\mathrm{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\mathrm{pub}(B)$

Payment protocol

The terminal T checks the authenticity of the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathsf{priv}(B)}$$

Payment protocol

The terminal T checks the authenticity of the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathrm{priv}(B)}$$

The terminal T asks the code:

$$\begin{array}{lccll} 2. & T & \rightarrow & Cl : & PIN? \\ 3. & Cl & \rightarrow & C : & 1234 \\ 4. & C & \rightarrow & T : & ok \end{array}$$

Payment protocol

The terminal T checks the authenticity of the credit card C:

$$1.\quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathsf{priv}(B)}$$

The terminal T asks the code:

$$\begin{array}{lcccl} 2. & T & \rightarrow & Cl: & PIN? \\ 3. & Cl & \rightarrow & C: & 1234 \\ 4. & C & \rightarrow & T: & ok \end{array}$$

The terminal T requests authorisation from the bank B:

$$\begin{array}{lcccl} 5. & T & \rightarrow & B: & auth? \\ 6. & B & \rightarrow & T: & 4528965874123 \\ 7. & T & \rightarrow & C: & 4528965874123 \\ 8. & C & \rightarrow & T: & \{4528965874123\}_{K_{CB}} \\ 9. & T & \rightarrow & B: & \{4528965874123\}_{K_{CB}} \\ 10. & B & \rightarrow & T: & ok \end{array}$$

Attack against credit cards

Initially, security was guaranteed by:

- cards hard to replicate,
- secrecy of keys and protocol.

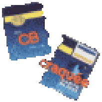
Attack against credit cards

Initially, security was guaranteed by:

- cards hard to replicate,
- secrecy of keys and protocol.

However, there are attacks!

- cryptographic attack: 320-bit keys are no longer secure,
- logical attack: no link between PIN and authentication,
- hardware attack: replication of cards.



→ "YesCard" made by Serge Humpich in 1997

The "YesCard": how does it work?

Logical attack

$$
\begin{array}{llll}
1. C & \rightarrow T & : \mathsf{Data}, \{\mathsf{hash}(\mathsf{Data})\}_{\mathsf{priv}(B)} \\
2. T & \rightarrow CI & : PIN? \\
3. CI & \rightarrow C & : 1234 \\
4. C & \rightarrow T & : ok
\end{array}
$$

The "YesCard": how does it work?

Logical attack

$$
\begin{array}{llll}
1. C & \rightarrow T & : \text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)} \\
2. T & \rightarrow CI & : PIN? \\
3. CI & \rightarrow C' & : 2345 \\
4. C' & \rightarrow T & : ok
\end{array}
$$

The "YesCard": how does it work?

Logical attack

$$
\begin{aligned}
1.\,C &\quad \rightarrow T &&: \text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)}\\
2.\,T &\quad \rightarrow CI &&: PIN?\\
3.\,CI &\quad \rightarrow C' &&: 2345\\
4.\,C' &\quad \rightarrow T &&: ok
\end{aligned}
$$

$\rightarrow$ copy card data and certificate on card that accepts all PINs

The "YesCard": how does it work?

Logical attack

$$\begin{array}{llll} 1.C & \rightarrow T & : \text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)} \\ 2.T & \rightarrow CI & : PIN? \\ 3.CI & \rightarrow C' & : 2345 \\ 4.C' & \rightarrow T & : ok \end{array}$$

$\rightarrow$ copy card data and certificate on card that accepts all PINs

$$\begin{array}{llll} 1.C' & \rightarrow T & : \text{XXX}, \{\text{hash}(\text{XXX})\}_{\text{priv}(B)} \\ 2.T & \rightarrow CI & : PIN? \\ 3.CI & \rightarrow C' & : 0000 \\ 4.C' & \rightarrow T & : ok \end{array}$$

Credit cards: going further

Preventing the YesCard attack

Credit cards: going further

Preventing the YesCard attack

Flaw fixed with the more recent Dynamic Data Authentication mode.
Each card has signature key pair (whose public key is authentified) used to sign Data together with a random value (challenge).

Credit cards: going further

Preventing the YesCard attack

Flaw fixed with the more recent Dynamic Data Authentication mode. Each card has signature key pair (whose public key is authentified) used to sign Data together with a random value (challenge).

New "attacks": contact-less cards

The same protocol is used through wireless communications:

$\rightsquigarrow$ Card data easily harvested!

Included card holder name and record of purchase in France until 2013.

Conclusion: security not the main goal of credit card consortium.

Formal Methods

The problem

In summary:

- Protocol design is error prone.
  Hard to clearly define threats and security properties.
- Flaws undetected by testing appear in presence of adversary.
- Errors can have serious consequences.

$\rightsquigarrow$ Formal methods and formal proofs are needed!

Active and successful research for several decades.

Models of protocols: communication assumptions

Attacker capabilities

- The attacker can intercept all messages sent on the network.
- He can compute messages using crypto primitives.
- He can send messages on the network.

A worst case scenario where

the attacker is (perhaps maliciously) executing the protocol.

Models of protocols: the computational model

The computational model has been developed in the early 80'
by Goldwasser, Micali, Rivest, Yao, and others.

- Messages are bitstrings.
- Cryptographic primitives are computations on bitstrings.
- The attacker is any probabilistic (polynomial-time) Turing machine.
- Cryptographic assumptions on primitives: examples for encryption, hash functions, signatures.

Realistic but (until recently) does not lend itself to fully formal proofs.

Models of protocols: the formal model

The formal model, aka. symbolic or "Dolev-Yao model"
is due to Needham and Schroeder [1978] and Dolev and Yao [1983].

- Cryptographic primitives are blackboxes.
- Messages are terms on these primitives.
  $\hookrightarrow$ $\{m\}_k$ encryption of the message m with key k,
  $\hookrightarrow$ $\langle m_1, m_2 \rangle$ pairing of messages m_1 and m_2, ...
- The attacker is restricted to compute only using these primitives, according to some equations.
  $\hookrightarrow$ $\mathrm{dec}(\{m\}_k, k) = m$
  $\hookrightarrow$ $\mathrm{fst}(\langle x, y \rangle) = x$
  $\Rightarrow$ perfect cryptography assumption

Models of protocols: the formal model

The formal model, aka. symbolic or "Dolev-Yao model"
is due to Needham and Schroeder [1978] and Dolev and Yao [1983].

- Cryptographic primitives are blackboxes.
- Messages are terms on these primitives.
  $\hookrightarrow$ $\{m\}_k$ encryption of the message m with key k,
  $\hookrightarrow$ $\langle m_1, m_2 \rangle$ pairing of messages m_1 and m_2, ...
- The attacker is restricted to compute only using these primitives, according to some equations.
  $\hookrightarrow$ $\mathrm{dec}(\{m\}_k, k) = m$
  $\hookrightarrow$ $\mathrm{fst}(\langle x, y \rangle) = x$
  $\Rightarrow$ perfect cryptography assumption

Lends itself well to automatic proofs: AVISPA, ProVerif, ...
Delicate hypothesis: the only equalities are those given by the equations.

Relationship between the two models

An attack in the formal model is an attack in the computational model.
What more can we say?

Relationship between the two models

An attack in the formal model is an attack in the computational model.
What more can we say?

Computational soundness (under very specific assumptions)

Proof in the formal model $\Rightarrow$ proof in the computational model

Approach pioneered by Abadi&Rogaway [2000]; many works since then.

Relationship between the two models

An attack in the formal model is an attack in the computational model.
What more can we say?

Computational soundness (under very specific assumptions)

Proof in the formal model $\Rightarrow$ proof in the computational model

Approach pioneered by Abadi&Rogaway [2000]; many works since then.

The computational model is still a model:

beware side channel attacks and implementation flaws!

Verifying protocols in the formal model

The set of all terms that the attacker can obtain is infinite:

- The attacker can generate messages of unbounded size.
- The number of sessions of the protocol is unbounded.

Verifying protocols in the formal model

The set of all terms that the attacker can obtain is infinite:

- The attacker can generate messages of unbounded size.
- The number of sessions of the protocol is unbounded.

Solutions:

- Bounded messages and number of sessions
  $\Rightarrow$ finite state model checking: FDR [Lowe, TACAS'96]

Verifying protocols in the formal model

The set of all terms that the attacker can obtain is infinite:

- The attacker can generate messages of unbounded size.
- The number of sessions of the protocol is unbounded.

Solutions:

- Bounded messages and number of sessions
  $\Rightarrow$ finite state model checking: FDR [Lowe, TACAS'96]
- Bounded number of sessions but unbounded messages
  $\Rightarrow$ constraint solving: Cl-AtSe, integrated in AVISPA

Verifying protocols in the formal model

The set of all terms that the attacker can obtain is infinite:

- The attacker can generate messages of unbounded size.
- The number of sessions of the protocol is unbounded.

Solutions:

- Bounded messages and number of sessions
  $\Rightarrow$ finite state model checking: FDR [Lowe, TACAS'96]
- Bounded number of sessions but unbounded messages
  $\Rightarrow$ constraint solving: Cl-AtSe, integrated in AVISPA
- Unbounded messages and number of sessions $\Rightarrow$ undecidable
  - Interactive theorem proving: Isabelle [Paulson, JCS'98]
  - Approximations:
    - abstract interpretation [Monniaux'03], TA4SP integrated in AVISPA
    - typing [Abadi'99], [Gordon & Jeffrey '02]
  - Semi-decision procedures (and approximations): Proverif

Proofs in the computational model

Proof by sequence of game reductions [Shoup, Bellare & Rogaway].
Games correspond to security property of protocol or primitive,
or mathematical assumption.

Proofs in the computational model

Proof by sequence of game reductions [Shoup, Bellare & Rogaway].
Games correspond to security property of protocol or primitive,
or mathematical assumption.

Example: IND-CPA

- Generate public and private keys.
- Allow attacker to (polynomially) perform encryptions.
- Attacker chooses m_0, m_1.
- Challenger chooses $i \in \{0, 1\}$, shows $\mathrm{enc}(m_0)$.
- Attacker should not be able to guess i with probability much better than $1/2$.

Proofs in the computational model

Proof by sequence of game reductions [Shoup, Bellare & Rogaway].
Games correspond to security property of protocol or primitive,
or mathematical assumption.

Example: IND-CPA

- Generate public and private keys.
- Allow attacker to (polynomially) perform encryptions.
- Attacker chooses m_0, m_1.
- Challenger chooses $i \in \{0, 1\}$, shows $\mathrm{enc}(m_0)$.
- Attacker should not be able to guess i with probability much better than $1/2$.

Automation

CryptoVerif, Certicrypt, F$\star$...

State of the art

A growing list of successes:

1996 Needham-Schroeder public-key protocol attack, 17 years later

2008 Google's non-compliant SSO implem. attack (AVISPA)

2010 Freak: trick SSL implems to choose export crypto (MiTLS)

2016 Proofs of TLS (MiTLS, Tamarin)

2016 Proof of BAC e-passport protocol (UKano/Proverif)

Some industrial acceptation:

- Involvement of academic community in the design of TLS 1.3
- Some collaborations around 5G protocol design.
- Mozilla has incorporated certified crypto implementations into its library.

Conclusion

Difficulties: a big picture

- Communicating agents: concurrent, bounded or not
- Crypto primitives: symbolic vs. computational, algebraic properties
- Hostile environment: passive vs. active, deduction, equivalence

Upcoming lectures

- Verification in the symbolic model (D. Baelde, 8 sessions)
- Mechanizing computational security proofs (B. Blanchet, 4 sessions)
- Verification of protocol implementations (K. Bhargavan, 4 sessions)

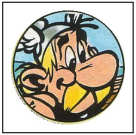
Conclusion

Upcoming lectures

- Verification in the symbolic model (D. Baelde, 8 sessions)
  - Computational and symbolic models (1/8)
  - Verification in the symbolic model (5/8)
    - The deduction problem
    - Symbolic model-checking
    - Indistinguishability
    - Proverif: the unbounded case, correspondences, diff-equiv.
  - Computationally complete symbolic attacker (1/8)
- Mechanizing computational security proofs (B. Blanchet, 4 sessions)
- Verification of protocol implementations (K. Bhargavan, 4 sessions)

Exercises

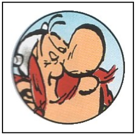
Needham-Schroeder's Protocol (1978)



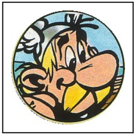
- $$\begin{array}{lcll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Needham-Schroeder's Protocol (1978)

$$\begin{array}{lcccr} & A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ \bullet & B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ & A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$



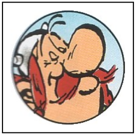
Needham-Schroeder's Protocol (1978)



$$
\begin{array}{lcll}
 & A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\
 & B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
\bullet & A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Needham-Schroeder's Protocol (1978)

$$\begin{array}{cccr} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$



Needham-Schroeder's Protocol (1978)

$$\begin{array}{rcll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Questions

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\mathrm{pub}(B)}$, does it really come from A ?

Needham-Schroeder's Protocol (1978)

$$\begin{array}{rcll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Questions

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\mathrm{pub}(B)}$, does it really come from A ?

Attack

An attack was found 17 years after its publication! [Lowe 96]

Example: Man in the middle attack

Agent A

Intruder I

Agent B

Attack

- involving 2 sessions in parallel,
- an honest agent has to initiate a session with I.

$$\begin{array}{lcl} A \to B & : & \{A, N_a\}_{\mathrm{pub}(B)} \\ B \to A & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A \to B & : & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Example: Man in the middle attack

Agent A

$$\xrightarrow{\{A, N_a\}_{\mathsf{pub}(I)}}$$

Intruder I

$$\xrightarrow{\{A, N_a\}_{\mathsf{pub}(B)}}$$

Agent B

$$
\begin{array}{lcl}
\mathsf{A} \rightarrow \mathsf{B} & : & \{A, N_a\}_{\mathsf{pub}(B)} \\
\mathsf{B} \rightarrow \mathsf{A} & : & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
\mathsf{A} \rightarrow \mathsf{B} & : & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

Example: Man in the middle attack

$\{A, N_a\}_{\mathrm{pub}(I)} \longrightarrow$

$\longleftarrow \{N_a, N_b\}_{\mathrm{pub}(A)}$

$\{A, N_a\}_{\mathrm{pub}(B)} \longrightarrow$

$\longleftarrow \{N_a, N_b\}_{\mathrm{pub}(A)}$

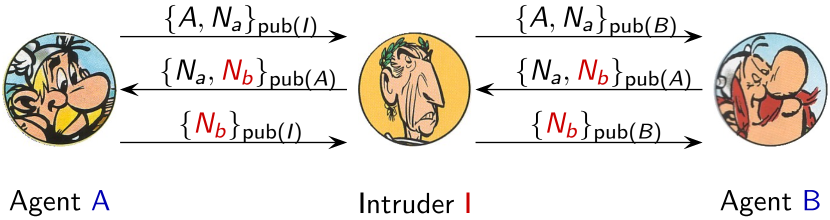
Agent A

Intruder I

Agent B

$$
\begin{array}{lcl}
\mathrm{A} \rightarrow \mathrm{B} & : & \{A, N_a\}_{\mathrm{pub}(B)} \\
\mathrm{B} \rightarrow \mathrm{A} & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
\mathrm{A} \rightarrow \mathrm{B} & : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Example: Man in the middle attack



$$\begin{array}{lcl} \mathrm{A} \to \mathrm{B} & : & \{A, N_a\}_{\mathrm{pub}(B)} \\ \mathrm{B} \to \mathrm{A} & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ \mathrm{A} \to \mathrm{B} & : & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Example: Man in the middle attack

$\{A, N_a\}_{\mathrm{pub}(I)} \longrightarrow$

$\longleftarrow \{N_a, N_b\}_{\mathrm{pub}(A)}$

$\{N_b\}_{\mathrm{pub}(I)} \longrightarrow$

$\{A, N_a\}_{\mathrm{pub}(B)} \longrightarrow$

$\longleftarrow \{N_a, N_b\}_{\mathrm{pub}(A)}$

$\{N_b\}_{\mathrm{pub}(B)} \longrightarrow$

Agent A — Intruder I — Agent B

Attack

- The intruder knows N_b.
- When B finishes his session (apparently with A), A has never talked with B.

$$
\begin{array}{lcl}
A \rightarrow B & : & \{A, N_a\}_{\mathrm{pub}(B)} \\
B \rightarrow A & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A \rightarrow B & : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Exercise: Needham-Schroder

$$
\begin{array}{lcl}
\mathrm{A} \rightarrow \mathrm{B} & : & \{A, N_a\}_{\mathrm{pub}(B)} \\
\mathrm{B} \rightarrow \mathrm{A} & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
\mathrm{A} \rightarrow \mathrm{B} & : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Question

Propose a fix to ensure mutual authentication.

Exercise: LAK

RFID tags must be authenticated by a reader.
Each tag owns a secret key k, shared with the reader.

$$
\begin{array}{lcl}
R & \rightarrow & T & : & n_R \\
T & \rightarrow & R & : & n_T, h(n_R \oplus n_T \oplus k) \\
R & \rightarrow & T & : & h(h(n_R \oplus n_T \oplus k) \oplus k \oplus n_R)
\end{array}
$$

Questions

- Show that the reader does not properly authenticate the tag. It is enough to consider a scenario with one tag and one reader, with the same shared secret k.
- Propose a fix.

Exercise: Private Authentication

A and B each have a secret key sk_X and a public key pk_X.

$$\begin{array}{ccccc} A & \rightarrow & B & : & \{n_A, pk_A\}_{pk_B} \\ B & \rightarrow & A & : & \{n_A, n_B, pk_B\}_{pk_A} \end{array}$$

Question

Can an attacker distinguish between:

- a role B willing to talk to A, and
- a role B willing to talk to A' ?